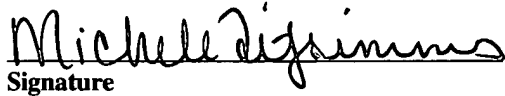


I hereby certify that this paper and/or fee is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 CFR §1.10 on the date indicated below and is addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

  
Signature

DATE OF DEPOSIT: December 11, 2003

EXPRESS MAIL LABEL NO.: EV331727159US

Inventor[s]: David C. Challener, Steven J. Mastrianni, Joseph R. Parker, Ratan Ray, and Leendert P. Van Doom

## COMPUTER PRODUCT AND SYSTEM FOR ESTABLISHING NETWORK CONNECTIONS

### FIELD OF THE INVENTION

The present invention relates to computer network systems and in particular to a  
5 computer product and system for establishing a network connection between a client system  
and the network.

### CROSS REFERENCE TO RELATED APPLICATIONS

The subject matter of this application is related to two (2) U.S. patent application  
10 numbers \_\_\_\_\_ and \_\_\_\_\_ filed December 11, 2003 as attorney docket numbers  
RPS920030249US1 and RPS920030250US1, the teachings of which are incorporated herein  
by reference.

### BACKGROUND OF THE INVENTION

15 Despite advances in hardware and software technology, computer users frequently  
encounter situations in which they cannot establish a connection to a service provider or  
network. What is meant by establishing a connection in the context of the present application

is either creating a first connection to a network or repairing a connection to a network.

Repairing a connection to a network can be the result of an improperly installed or configured

software program or device driver, or the device being used to attempt the connection may

simply be disabled. In the case of a wired connection, the problem may be caused by a

defective interconnect cable, or in the case of a wireless connection, a bad transmitter or

antenna connection. In addition to such defects, other problems in the network can prevent the

user from getting connected. For example, in the case of an Ethernet network, the failure of the

network's Dynamic Host Configuration Protocol, or DHCP server, can prevent the user from

getting connected to the network.

Because the majority of these problems are not communicated to the user, the user

cannot determine the cause, and sees the problem as simply "no connection." Moreover, even

if the user was able to determine the source of the problem, e.g., through an error message

generated by the system, he or she would most likely not know how to resolve the problem.

Accordingly, what is needed is a system and method for determining the cause of a

connectivity problem and repairing the connection. The system and method should be

automatic and transparent to the user. The present invention addresses such a need.

## SUMMARY OF THE INVENTION

A method, computer readable medium and computer system for establishing a network

connection between a client system and a network is disclosed. In a first aspect, the method

preferably includes collecting real time connectivity information by the client system and

utilizing the real time connectivity information by the client system to establish a connection

with the network. In a second aspect, a computer system coupled to a network includes at least

one network adapter for monitoring and collecting real time connectivity information from the

network, memory for storing the real time connectivity information, and a processor coupled to the memory and to the at least one network adapter, where the processor is configured to execute program instructions for utilizing the real time connectivity information to repair a failed network connection between the computer system and the network.

5

## **BRIEF DESCRIPTION OF THE DRAWINGS**

Figure 1 illustrates a block diagram of a system configuration for a preferred embodiment of the present invention.

Figure 2 is a block diagram of a client system according to a preferred embodiment of the present invention.

10

Figure 3 illustrates a logical software block diagram of the preferred embodiment of the present invention.

Figure 4 is a flowchart illustrating a process for establishing a network connection according to a preferred embodiment of the present invention.

15

## **DETAILED DESCRIPTION**

The present invention relates to computer network systems, and in particular to a method and system for establishing a network connection between a client system and the network. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded

20

the widest scope consistent with the principles and features described herein.

The preferred embodiment of the present invention determines the cause of a connectivity problem and attempts to repair the connection by applying changes to the operating software and/or device configuration in an iterative fashion until a connection is established. It determines the types of changes or repairs to make by collecting information related to the network and to the status of system components, and based in part on that information, begins an iterative process of attempting to establish the connection. The system and method according to a preferred embodiment of the present invention includes an inference engine that determines the cause of the connection failure and the appropriate corrective action based on information contained in several sources, including but not limited to, a local persistent knowledgebase, a real time network list, a set of local rules, and a server-resident database. In a preferred embodiment, the system and method of the present invention evaluates the conditions of a particular connection, and by consulting one or more of the above listed sources, determines a hierarchy of potential solutions and applies each potential solution iteratively until a connection is established. The result of each iterative attempt is monitored, and if unsuccessful, the result, e.g., an error message, is used as feedback to the inference engine.

Figure 1 illustrates a block diagram of a system configuration for a preferred embodiment of the present invention. Preferably the present invention is implemented on a client computer systems 100a and/or 100b. As is shown, a first client computer system 100a is coupled to a public network 120, such as the Internet. A second client computer system 100b is coupled to a private network 110 such as a Local Area Network (LAN). The private network 110 is coupled to the public network 120 via a gateway 103. Nonetheless, those

skilled in the art appreciate that a client system 100 (i.e., either client computer system 100a or client computer system 100b) can be coupled to either a private or public network, and not necessarily to both. The client computer system 100 can be mobile, e.g., a laptop or handheld personal computer, or a stationary desktop. A user uses the client computer system 100 to perform information management tasks, including sending and receiving electronic mail from a mail server 140 or from a company server 112, retrieving web pages from a web server 150, and sending and receiving data files from a file server 130 or the company server 112. The client 100 includes an operating system and appropriate hardware adapters such as a dial-up modem or wireless card, or a network adapter such as Token Ring or Ethernet that allows connection to a network 110, 120 through a cable modem, DSL modem, hub, or switch.

Figure 2 is a block diagram of the client computer system 200 according to a preferred embodiment of the present invention. The client 200 includes a processor or central processing unit (CPU) 212 that is coupled to memory 214 (e.g., system, RAM, ROM), a display device 216, input/output devices 218, and two network adapters 219 and 219a for connecting the client 200 to the network 220, e.g., via a wired or wireless connection, respectively. In the preferred embodiment, the network adapters, e.g., 219 and 219a, are capable of monitoring and capturing network traffic in real time using a wired or wireless method, as is well known in the art.

Figure 3 illustrates a logical software block diagram of the preferred embodiment of the present invention 300. As is shown, the preferred embodiment of the present invention includes an inference engine 302, a verify function 304 and a connection manager 306. The inference engine 302 provides functionality for automatically determining the cause of a failed connection and for generating a hierarchy of solutions to repair the connection to the network

220.

In a preferred embodiment, the inference engine 302 analyzes one or more error messages 316 generated by the client 200 relating to the failed connection to assist it in determining the cause of the connectivity failure. In addition, the inference engine 302 invokes the verify function 304, which audits each of the communication devices to determine which, if any, can be potential candidates for connectivity. Based on the information received from the verify function 304 and on its analysis of the error messages 316, the inference engine 102 determines the cause of the connectivity failure.

The inference engine 302 utilizes connectivity information stored in the client computer system 200 to repair the connection based on its diagnosis of the cause for failure. The connectivity information includes a set of local rules or preferences 308, a local persistent knowledgebase 310 a real time network list 314 and optionally, a remote/server-resident knowledgebase. The local rules 308 indicate the client's 200 connection preferences. For example, if the client 200 is capable of establishing a wired and wireless connection, but prefers a wireless connection, the inference engine 302 will attempt to establish a wireless connection before other modes of connectivity. The local persistent knowledgebase 310 includes static configuration information, e.g., parameters and settings. The local persistent knowledgebase 310 can be downloaded and/or updated from the remote/server-resident knowledgebase 312 stored on the company server 112 (Figure 1).

According to a preferred embodiment of the present invention, the real time network list 314 is a weighted list that includes connectivity information gathered by the network adapter 219 (Figure 2). As stated above, the network adapter 219 monitors all network traffic, not just the traffic directed toward the client 200, and collects certain connectivity information

in real time. Such information includes addresses of DHCP servers 114, DNS servers 160 and gateways, addresses and names of SOCKS servers, names and addresses of printers, IP addresses recently assigned by the DHCP server 114, and other connectivity information. The list 314 is weighted such that the most popular, i.e. most utilized, addresses appear highest on the list 314.

With the connectivity information, the inference engine 302 formulates a best solution, which is then passed to the connection manager 306 and implemented, i.e., a network connection is attempted using the solution. If the solution fails, such information is transmitted back to the inference engine 302, e.g., via an error message 316, so that a new diagnosis of the connection failure can be generated if necessary. This process repeats until a connection is established.

Figure 4 is a flowchart illustrating a process for establishing a network connection according to a preferred embodiment of the present invention. Referring to Figures 1-4 together, the process begins at step 402, where connectivity information is collected and stored in memory 214 (Figure 2). This step may include downloading or updating the static configuration information from the remote/server-resident knowledgebase 312 in the company server 112 for the local persistent knowledgebase 310, and monitoring and collecting connectivity information from the network 220 via the network adapter 219 for the real time network list 314. In step 403, the connectivity information is utilized to make a connection. In step 404, it is determined if a connection failure is detected by the client 200.

If a connection error is determined then an error message is transmitted and in step 406, the root cause of the connection failure is determined. In a preferred embodiment, the inference engine 302 is called automatically once a connection failure is detected. In another

embodiment, the user can invoke the inference engine 302. In any event, once the inference engine 302 is called, the inference engine 302 invokes the verify function 304. The verify function 304 audits each communication device to determine its status, e.g., functional or failed, thereby determining which of them are potential candidates for connectivity. The results of the audit are returned to the inference engine 302, which then analyzes the results and the error message(s) 316 in order to determine the root cause of the connection failure.

Once the root cause has been determined, the inference engine 102 generates a best solution (in step 408) using the connectivity information based on the root cause. For example, in one case, the inference engine 302 monitors the range of IP addresses that are assigned by a DHCP server 114. It then selects an address in the range and instructs the connection manager 306 to “ping” that address to determine if it is in use. If the address is not in use, the inference engine 302 temporarily assigns the IP address to the client 200 and sets up its network settings for that connection. In another example, if the inference engine 302 determines that the root cause of a connection failure is due to a missing field, e.g., an IP address for the DHCP server 114 or DNS server 160, from the current discovered configuration, the inference engine 302 will insert the appropriate IP address from the real time network list 314, effectively “filling in the blanks.” Because the real time network list 314 is a weighted list, the inference engine 302 applies the most frequently utilized IP addresses, which are also those most likely to succeed.

In step 410, the connection manager 306 implements the best solution . If the connection is unsuccessful (step 412), i.e., the best solution fails, then that result is passed back to the inference engine 302 which reexamines its diagnosis based, in part, on the previous unsuccessful attempt and generates a next best solution (step 414). The process ends when the connection is successful or when all potential solutions have been exhausted.



In summary, the preferred embodiment of the present invention automatically determines the root cause of a connection failure and attempts to repair the connection without intervention from a user. To do this, the inference engine 302 is invoked to analyze one or more error messages 316 related to the connection failure to determine the root cause of the failure. Once the cause is determined, the inference engine 302 utilizes connectivity information stored in the client 200 to repair the connection. The connectivity information includes real time network information, e.g., IP addresses of the DHCP servers and domain name servers, collected by the client's network adapter 219. The real time network information is stored in a weighted list 314, with the most frequently assigned addresses at the top. By utilizing the preferred embodiment of the present invention, the client 200 seamlessly and transparently repairs a failed network connection.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.